304234

UNLIMITED
AD-A241 821

2

RSRE
MEMORANDUM No. 4504

ROYAL SIGNALS & RADAR ESTABLISHMENT

NOTES ON THE POLYINSTANTIATION PROBLEM

Author: S Wiseman

DTIC
S ELECTE D
OCT 16 1991
D

This document has been approved for public release and sale; its distribution is unlimited.

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

91-13258

RSRE MEMORANDUM No. 4504

UNLIMITED

91 [illegible]

CONDITIONS OF RELEASE

0107730 304234

.... DRIC U

COPYRIGHT (c)
1988
CONTROLLER
HMSO LONDON

.... DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4504

Title:	Notes on the Polyinstantiation Problem
Author:	Simon Wiseman
Date:	July 1991

INSPECTED

Summary

This memorandum collects together three working notes concerning the database security technique called polyinstantiation The first is the position paper given for a panel session held at the Computer Security Foundations Workshop in June 1991 This outlines the relationship between confidentiality and integrity in general terms The second gives the more specific position relating to the problems of polyinstantiation and how they are avoided with the insert low approach The third note gives an example which illustrates the difference in attitude towards integrity between polyinstantiation and the insert low approaches

By
Dist
A-1

Copyright
©
Controller HMSO London
1991

INTENTIONALLY BLANK

Contents

INTENTIONALLY BLANK

The Conflict between Confidentiality and Integrity

Published in the Proceedings of Computer Security Foundations Workshop IV,
Franconia NH, June 1991,
IEEE Computer Society, pp241-242.

1. Integrity and Confidentiality

The purpose of security controls in a system is to offer the owners of the information held by the system a guarantee that the information is used and maintained properly. There are many mechanisms which are used to provide such guarantees, but these can be grouped into two main classes.

The first class are mechanisms which place defensive checks on the users of the system. These may be state based, for example "all salaries must be greater than $5000", or transition based, for example "no transaction can ever change a salary by more than $1000". This kind of check forms the first line of defence against stupid errors. However defensive checks offer no guarantee that the system's state is what it should be, only that it is not outrageously wrong.

The second class of security mechanism are those which vote on the outcome of a transition. Separation of duty rules come into this category, for example, "launch the missiles only if three independent authorities think its a good idea" Another example is "a user can read a document only if the originator and a reviewer agree that its ok".

Very often it is impractical to collect the vote of all those users who are required to sanction a transition. However, it is often possible for a user to automate their voting pattern, with statements like "I'll always agree to purchases of under $100, as long as you don't spend more than $1000 in one year". In fact, defensive checks can be thought of as automated votes cast by the system's owners, because such checks are effectively conditions imposed when authority is delegated.

A particularly interesting form of automated voting are confidentiality controls A simple rule would be that "information may be used only if the security officer agrees". However, since the security officers are too busy to vote on all transitions, a proxy rule is employed to automate their vote. For example, "I'll agree to any transition where sources of information are classified lower than the sinks" A security officer will, of course, also agree to other kinds of transition, for example when downgrading information, but only if involved personally

This then is the relationship between confidentiality controls, integrity controls and security There are two kinds of integrity control, defensive checks and voting, and confidentiality controls are an example of automated voting

2. Classifying Existence

The use of defensive checks leads to information about the state being given to the user This is in the form of an "ok" or "not ok" response. Thus any part of the state which is examined to evaluate the check becomes a source in terms of the confidentiality controls. An example of a defensive check is a uniqueness constraint, which is where duplicates are not allowed in some sequence of values Such constraints are used frequently in databases

The extra information flow needed to evaluate a check may mean that the automated voting rule no longer applies In the absence of full "manual" agreement being reached, the transition cannot proceed That is, a transition request is rejected because no agreement was reached that the evaluation of the defensive checks constituted proper use of the constrained information.

This in itself may not seem a great problem, it is just that once information classified at different levels is constrained in some way, users with low clearances will be unable to alter the low information. In fact the problem only becomes serious when one takes a more realistic view of confidentiality and considers classifying the existence of things.

Suppose then that the existence of some of the constrained information is highly classified. Now users with low clearances can never alter low information, because they are unable to determine whether it is possible for them to fully evaluate the check.

It is clear, therefore, that a practical system must strive to keep the existence of things classified as low as possible, and must impose constraints on this. This is so that users with low clearances can establish when a constraint can be fully evaluated.

3. Polyinstantiation

In operating systems things like files are created and destroyed. Programs usually create files because they wish to store data in them and so most operating systems will classify the existence of a file equal to the classification of its contents. A program does not need to be able to create a file whose contents are classified higher than its existence, because there is nothing very interesting about files you can detect but not use.

In databases things like tables, rows and fields are created and destroyed, however here things you can detect but not use are of value This is because relationships between things are interesting. For example, a low user may be allowed to know that a flight has some cargo but not what the cargo is That is the existence of the cargo field is classified low while its value is high. This can be useful, for example because the low user could be responsible for deleting the information when the flight arrives at its destination. In a database this can be done without revealing the nature of the cargo

Polyinstantiating databases have been conceived with the notions of secure operating systems in mind. In particular the existence of something, either a row or a field, is classified equal to the classification of its value. This means a user cannot determine whether a defensive check, such as uniqueness of "key" values in a table, can be properly evaluated Rather than prevent low users from modifying the database, the alternative, of banning important defensive checks, is invariably adopted

4. SWORD

SWORD has been designed with databases in mind. Here the existence and content of a field are clearly separated notions which can be classified separately As a result important defensive checks can be applied without preventing low users carrying out their work

In general a user may receive one of four types of reply If it is possible that some of the constrained fields are hidden, the reply is "not sure whether check can be properly evaluated". If it is known that all the fields can be detected, but some of their contents are highly classified, the reply is "cannot fully evaluate check". If all fields may be detected and examined the reply is either "check not satisfied" or "ok", depending on their value

5. Conclusion

It is a popular misconception that polyinstantiation is unavoidable in databases. Actually it is only unavoidable when an operating system's security controls are used to build a secure database. So, given a secure operating system, my reply to the question "How do I get a secure database?" would be "I wouldn't start from here".

INTENTIONALLY BLANK

Polyinstantiation *vs.* SWORD

Originally internal working note SRW/16/91, 12th July 1991

1. Introduction

A panel session on polyinstantiation was held at the Franconia Computer Security Foundations Workshop, June 1991. My position paper [Wiseman91a], which appears in the proceedings, gave my overall position with regards the conflict between confidentiality and integrity. However, the presentation I gave concentrated on my views on polyinstantiation and how this related to *SWORD* (the 'history' of *SWORD*'s development can be followed in [Wiseman*et.al.*89], [Wiseman89], [Wood90], [Wiseman90], [Wiseman91b], [Lewis91] and [Wood*et.al.*91]). This note records these specific comments Another note (SRW/19/91) records an example developed during the panel discussions.

2. What is Polyinstantiation?

"What is polyinstantiation?". One answer is that polyinstantiation is a thin veneer spread on top of multiple single-level tables, which gives the impression of multi-level security. In fact the result is not generally multi-level because the classification of something's existence must always equal the classification of that thing's contents

Polyinstantiation is not an inherent, intrinsic or inevitable property of the multi-level world, as claimed in many papers, eg. [Lunt91] and [Denning*et.al.*87]. View based classification schemes do not need to polyinstantiate [Wilson88], and in fact there is already a product based on this technique [Knode&Hunt88]. That is not to say view based schemes are the answer to the world's problems, since in practice their classification schemes are too inflexible, but they do at least prove that polyinstantiation is not inherent or intrinsic

Another alternative to polyinstantiation is the insert low approach [Wiseman90] This solution says (roughly) that only those with the lowest clearance can insert rows into a table The approach allows the database to be generally multi-level, in that it does allow the existence of something to be classified less than the classification of its contents. It turns out that this is what makes the DBMS useable and not just consist of single level tables

The insert low approach has been adopted in the *SWORD* DBMS That *SWORD* preserves confidentiality can be argued easily, because it hinges on an object hierarchy in the same way that Multics does [Bell&LaPadula76]. It is more difficult to argue that *SWORD* is useful However, unlike polyinstantiating DBMSs, it can enforce elementary constraints (eg uniqueness) and has clean semantics (eg after an update the number of rows is the same).

Of course, in *SWORD* it is possible to have single-level tables, so the 'thin veneer' can always be implemented explicitly in the application to give the effect of polyinstantiation That is *SWORD* has more descriptive power than polyinstantiation, so if you don't accept *SWORD* how can you accept polyinstantiation?

Now let us ask "what is polyinstantiation?" again and try to establish a technical answer. Suppose you wish to build a database and you want the DBMS to enforce a uniqueness constraint for you (eg no two rows in the ship table have the same ship name) Now, for confidentiality reasons, you choose not to enforce this constraint (note that you can always enforce the constraint by not allowing anything to be inserted, so it is that you 'choose not to' rather than you 'cannot') Instead you get the DBMS to enforce a weaker constraint, namely that things are unique per classification. This whole process is polyinstantiation.

So the suggestion is that polyinstantiation is not to do with DBMS implementation mechanisms but with the database design process. Obviously an easy way to implement the 'unique per security level' constraint is to extend the unique name with its classification.

However, by this definition, making the name and its class unique does not necessarily mean that you are polyinstantiating, because this is one good way of partitioning a name space. For example, documents may be given serial numbers so that the numbering scheme for secret documents is separate from that for unclassified documents. So here the classification is really part of the name, not artificially so, hence there is no polyinstantiation.

3. Conclusion

We must ask "do we need polyinstantiation?" and if so "do we dare use it?". The work we have done with *SWORD* has shown that polyinstantiation can be avoided by using the insert low approach with appropriate database design techniques. The use of polyinstantiation will place the onus for integrity enforcement on each and every application This is not only expensive but could well have disastrous consequences [Wiseman89].

Finally, it is often said that polyinstantiation is necessary for cover stories. However, [Wiseman91b] discusses the importance of integrity in maintaining cover stories and shows how much better *SWORD* is than polyinstantiation for this task.

The Franconia Airfield Problem: A Moral Tale

Originally internal working note SRW/19/91, 18th July 1991.

1. Introduction

A panel session at the Franconia Computer Security Foundations Workshop held in June discussed the problem of polyinstantiation in databases. During the discussions I outlined an example, based on the airfield across the road from the Franconia Inn, which emphasised the differences of attitude to integrity between the 'pro' and 'anti' polyinstantiation camps. This note describes the example and the moral it contains.

2. The Scene

The Computer Security Foundations Workshop is held at the Franconia Inn, just outside the small town of Franconia in the White Mountains of New Hampshire. Across the road is a grass airstrip used by light aircraft and gliders.

For the purpose of this tale, it is assumed that the USAF are developing the technology to make aircraft invisible and have produced a prototype stealth glider, which by virtue of its silent running and invisibility can only be detected by touching it. Of course such technology is extremely sensitive, so much so that field trials are being conducted in Franconia under the 'cover story' of the research team's annual vacation.

Now it so happens that the stealth glider is being tested in June. The airfield is small and can accommodate only ten aircraft at the side of the runway, but currently there are eight normal aircraft plus the stealth glider, so there's really one free space although it looks like there are two. Obviously there is a constraint on the positions of the aircraft in that no two aircraft can be in the same space. This can be expressed as "the positions of all aircraft are unique".

June is when the Foundations Workshop is being held and this year two of the workshop's participants, Polly Instantiation and Ian Sertlow, are flying to Franconia in a light aircraft. Both are database security experts. Polly is an American and backs the idea of polyinstantiation, while Ian is British and is for the insert low approach

We shall now consider the many possible ways in which the details of the stealth project can be classified and assess the impact on Polly and Ian as they come into land and how the outcome depends upon who is at the controls.

3. Take One

Here the USAF classify everything they can which relates to their glider, because the project is so sensitive. Hence the existence of the 'unique position' constraint is highly classified, as is the existence of the stealth glider and all details about it, including its location.

In this case the consequences are the same regardless of whether it is Polly or Ian at the controls On approaching the airfield they would not know of the existence of the constraint and so would land and taxi to a space that looks free. Of course Murphy's Law says that the 'free' space is already occupied by the invisible glider and the resulting crash puts the project back several years.

Note that Polly and Ian would choose an apparently free space in the interests of self-preservation. It is rather obvious that you should not park in a space already occupied by another aircraft, so there is really no need for the airfield controllers to enforce such a constraint. However, its included because if a pilot was having an off day they might just do something foolish[1].

4. Take Two

This time the USAF admit to the existence of the constraint by making its existence unclassified, but they still keep its details highly classified. Thus both Polly and Ian are aware of something being constrained, but neither know what it is. In this case, Polly and Ian would take different courses of action.

Polly would approach the runway and notice that something is being constrained. Her attitude is "what the hell, it doesn't say its anything to do with me and I'm not breaching any confidentiality by going in". So Polly lands and disaster ensues.

Ian's attitude, as he approaches the runway, would display typical British Reserve: "something's constrained but I don't know what, maybe it would stop me landing, I'd better not try". So Ian would fly on and crash into the mountains.

5. Take Three

Rather than keep the details of the constraint a secret, the USAF now make it known to all. However, they still keep the existence of the stealth glider highly classified. So to Polly and Ian there appear to be two free spaces.

Both Polly and Ian would therefore land and choose what seems to be a free space, believing that they will not violate the constraint. Of course the 'free space' is bound to contain the stealth glider.

It is interesting to note that although the USAF have made more information available in this case, the result as far as Ian is concerned is worse. By revealing the nature of the constraint, but not the constrained information, Ian is now misled into thinking it is safe to park in what seems to be a free space.

6. Take Four

Now the USAF admit that an extra aircraft exists. That is the existence of a ninth aircraft is made unclassified, but details about it are still highly classified In particular the strange aircraft's position is highly classified

Polly would land, but her attitude means that she parks in the first space that seems free. The fact that she is not allowed to know the position of the ninth aircraft does not bother her. This is because "the positions of the aircraft are unique per security level, so what the hell". So inevitably, Murphy's Law strikes again.

Ian's cautious attitude, however, would mean that he dare not choose either of the seemingly free spaces to park in, just in case the one he chooses contains the mysterious ninth aircraft So he flys away in the hope of finding a more accommodating airfield.

7. Take Five

Finally the USAF admit to the position of their stealth glider, though all other details are kept highly classified Of course, in doing this they have really gone too far in revealing the nature of their project, but at least the airfield can be used safely Both

[1]This is an analogy of untrusted software that usually works OK, but sometimes goes wrong and chooses a space that's already in use

Polly and Ian are now able to land and find the free space, confident in the knowledge that it will not contain an invisible aircraft.

8. The Moral

This tale illustrates a number of points about classifying constraints if they are to be successfully enforced without preventing legal actions:

- the constraint must be detectable by those who are constrained;
- details of the constraint must be revealed to those who are constrained;
- the information mentioned in the constraint must all be detectable by those who are constrained;
- the information mentioned in the constraint must be visible to those who are constrained.

In terms of the example, then, it is necessary to reveal that:

- There is a constraint on the placement of aircraft;
- Each aircraft must occupy a different parking slot;
- There is an invisible aircraft;
- The invisible aircraft is in that slot.

9. Uniqueness Constraints in *SWORD*

The example just given used a uniqueness constraint as an example of what may happen when the existence and particulars of constraints are classified Now consider how the Moral Code derived from that tale applies to the enforcement of uniqueness constraints in *SWORD*.

In *SWORD* each row of a table has a label which gives the classification of that row's existence. Independently, each field of a row has a label which gives the classification of the field's contents A table also has a label which enforces an upper bound on the existence label of the rows within the table

When a new row is inserted, the row's existence label is set equal to the user's clearance An insert request fails if the user's clearance is not dominated by the table's row existence upper bound To enforce a "no flows down" confidentiality policy, a user is only allowed to detect those rows whose existence label is dominated by the user's clearance.

It is possible to specify that the values in a column are unique, that is no two rows have identical values in that column. In order to enforce such constraints when a new row is inserted, it is necessary that the user is able to inspect all the fields in the column.

From the first two points of the Moral Code given above it is clear that all users who are able to insert into a table must be able to detect and examine the column details of any uniqueness constraint applied to that table. In *SWORD* this is the case, because information about the constraint forms part of the schema and is visible to all users that are able to use the table

The third point of the Moral Code says that the user should be able to detect all the constrained information In the case of an insert in SWORD, however, this is only guaranteed to be true for those users whose clearance equals the row existence upper bound. Users with lower clearances would not be able to detect any rows which were inserted by users with higher clearance, and so cannot be sure that there is a row at the higher level Thus in order to ensure that the constraint is enforced, it is necessary to limit the ability to insert to those users whose clearance equals the row existence upper bound[1].

The fourth point of the Moral Code says that all the constrained information must be visible to the user. A field's value is only visible to a user if the user's clearance dominates the field's classification. So a user can only insert into a constrained table if no existing row has a field with a classification higher than their clearance in the constrained column.

This means that if a user inserts a row into a table, and they give a 'unique' field a classification higher than their clearance, no further inserts will be possible[2]. Of course, other integrity constraints will be applied to the database to prevent users causing denial of service problems in this way.

It must be noted that this behaviour is not an undesirable feature of *SWORD*, rather it is a direct consequence of the application designer's requirement for a uniqueness constraint. Techniques which seem to avoid this behaviour are in fact failing to enforce the uniqueness constraint as required by the designer.

10. Conclusion

There is a serious difference in the attitude to integrity constraints between the polyinstantiation and insert low approaches to database security. Polyinstantiation would rather that an integrity constraint was not enforced properly than some activity was prevented Insert low on the other hand would rather that activity is prevented than a constraint is violated.

The potential problem with the insert low approach, denial of service, can be fixed by appropriate data design, however with polyinstantiation the onus for integrity enforcement falls on the design and construction of the entire application software. The latter is not a sound economic proposition

[1]Note, this is a generalisation of the insert low approach which has been found to have some utility in the data dictionary

[2]Until the field is sanitised, the row deleted or the row existence upper bound is raised allowing users with higher clearances to insert

4. References[1]

Bell&LaPadula76
"Secure Computer System:
Unified Exposition and Multics Interpretation",
D.E.Bell & L.J.La Padula,
MITRE Corp. Report ESD-TR-75-306,
January 1976.

Denning*et.al.*87
"A Multilevel Relational Data Model",
D.E.Denning,T.F.Lunt,R.R.Schell,M.Heckman&W.Shockley,
Procs. IEEE Symp. Security and Privacy,
Oakland, CA, April 1987, pp220-234.

Knode&Hunt88
"Making Databases Secure with TRUDATA Technology",
R B.Knode & R.Hunt,
Procs. 4th Aerospace Comp. Sec. Applications Conf.,
Orlando, FL, December 1988, pp82-90

Lewis91 "The Front End Approach to Database Security",
S.R.Lewis,
Procs IFIP Sec'91,
Elsevier Advanced Technology,
Brighton, England, May 1991, pp443-454

Lunt91 "Polyinstantiation: an Inevitable Part of a Multilevel World',
T.F.Lunt,
Procs 4th Workshop on the Foundations of Computer Security,
Franconia, New Hampshire, June 1991, pp236-238.

Lunt&Hsieh91
"Update Semantics for a Multilevel Relational Database System",
T F Lunt & D Hsieh,
Procs IFIP WG11.3 Workshop on Database Security,
Halifax, UK, September 1990,
Published in Database Security IV - Status and Prospects,
S Jajodia & C E Landwehr (eds)
North Holland, 1991, pp 281-296

Wilson88 "Views as the Security Objects in a
Multilevel Secure Relational DBMS",
J. Wilson,
Procs IEEE Symp Security and Privacy,
Oakland, CA, April 1988, pp70-84.

[1]All RSRE Reports and Memoranda are freely available from
The Librarian, Defence Research Agency, RSRE Malvern, WORCS WR14 3PS
or the Author (email wiseman%uk mod hermes@uk mod relay)

Wiseman89 "On the Problem of Security in Data Bases",
S.R.Wiseman,
Procs. 3rd IFIP WG11.3 Working Conf. on Database Security,
Monterey, CA, September 198[illegible]
appears as: Database Securit[illegible] .. Status and Prospects,
D.Spooner & C.E.Landwehr (eds),
North-Holland 1990, ISBN 0 444 88701 6, pp301-310.

Wiseman90 "Control of Confidentiality in Databases",
S.Wiseman,
Computers and Security Journal, Vol 9 Num 6, October 1990, pp529-537.

Wiseman91a "The Conflict between Confidentiality and Integrity",
S.R.Wiseman,
Procs. 4th Workshop on the Foundations of Computer Security,
Franconia, New Hampshire, June 1991, pp241-242.

Wiseman91b "Lies, Damned Lies and Databases",
S.Wiseman,
RSRE Memo 4503, July 1991.

Wiseman*et.al.*89
"The Trouble with Secure Databases",
S.Wiseman, A.Wood & S.Lewis,
Procs MILCOMP '89,
Microwave Exhibitions and Publishers Limited,
London, September 1989, pp164-170

Wood90 "The SWORD Model of Multilevel Secure Databases",
A.W.Wood,
RSRE Report 90008,
June 1990.

Wood*et al* 91 "Achieving Confidentiality in Databases whilst Enforcing Uniqueness Constraints",
A W Wood, S.R Wiseman & S.R Lewis,
hopefully it'll appear somewhere this year!

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known) ____________

Overall security classification of sheet ____________UNCLASSIFIED____________

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No. MEMO 4504	Month JULY	Year 1991

Originators Name and Location RSRE, St Andrews Road Malvern, Worcs WR14 3PS
Monitoring Agency Name and Location
Title NOTES ON THE POLYINSTANTIATION PROBLEM

Report Security Classification UNCLASSIFIED	Title Classification (U, R, C or S) U

Foreign Language Title (in the case of translations)
Conference Details

Agency Reference	Contract Number and Period
Project Number	Other References
Authors WISEMAN, S	Pagination and Ref 11

Abstract

This memorandum collects together three working notes concerning the database security technique called polyinstantiation. The first is the position paper given for a panel session held at the Computer Security Foundations Workshop in June 1991. This outlines the relationship between confidentiality and integrity in general terms. The second gives the more specific position relating to the problems of polyinstantiation and how they are avoided with the insert low approach. The third note gives an example which illustrates the difference in attitude towards integrity between polyinstantiation and the insert low approaches.

Abstract Classification (U,R,C or S) U
Descriptors
Distribution Statement (Enter any limitations on the distribution of the document) UNLIMITED

S80/48

INTENTIONALLY BLANK